2022 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
DIGITAL ENGINEERING / SYSTEMS ENGINEERING TECHNICAL SESSION
AUGUST 16-18, 2022 - NOVI, MICHIGAN

Leveraging Game Engine Simulation to Accelerate Design

Andy Diepen[1], Orlando Vazquez[2], Andrew Black[3], Chuck Gaff[4]

[1]Chief Engineer, GS Engineering
[2] Head of Technology, Offworld Defence Simulations
[3]Lead Software Engineer, GS Engineering
[4] Tech Growth Manager, GS Engineering

ABSTRACT

Vehicle design today takes longer than it ever has in the past largely due to the abundance of requirements, standards, and new design techniques; this trend is not likely to change any time soon. This paper will explore how advancements in gaming engines can be leveraged to bring realistic visualization and virtual prototypes to the beginning of the design cycle, integrate subsystems earlier in the design, provide advanced simulation capabilities, and ensure that the final design not only meets the requirements but is fully vetted by stakeholders and meets the needs of the platform. The Unreal Engine and Bravo Framework can be used to bring this and more to vehicle designs to reduce design churn and bring better products to market faster.

Citation: A. Diepen, O. Vazquez, A. Black, C. Gaff "Leveraging Simulation Tools to Accelerate Design," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 16-18, 2022.

1. INTRODUCTION

In July 1940, invitations to bid were sent to 135 manufacturers for a light reconnaissance and command car or truck [1]. The invitation required that 70 vehicles be produced, the first due after 49 days, and the remainder within 75 days [2]. Three manufacturers stepped up to the plate, and although all were overweight and two manufacturers exceeded the deadline, it is amazing what was realized in such a short timeframe: the original Jeep. Today things are not as simple as they once were, and rapid prototyping has taken on a whole new meaning.

Engineering computing and software systems continue to get ever more complex, requiring vehicle-specific configurations that are tested, while data and cyber security are now a part of any military vehicle design. No development can be done without considering standards and regulatory requirements ranging from electromagnetic radiation to load ratings of tie-down and tow points. We now also have new technology which has migrated into suspension systems, driving aids, and user controls without even considering the new landscape of unmanned vehicles.

There is no question that vehicles today can do more and go farther all while providing a safer environment than vehicles 70 years ago; but all of this comes at a price of complexity,

significantly rising costs, not to mention the longer development and production times.

Vehicles used to be built with steel and hand tools, and we hear stories of vehicles being built out of spare parts. Today, to build means construction of model-specific tooling, procurement without waste, and software that must be configured and debugged before you can turn the key; or more appropriately, push the button.

Today, we continue to deal with the fallout from the COVID-19 pandemic as we all struggle with the challenges of a limited supply chain, and dramatically increasing material costs. Only making it harder to build a physical product than maybe any other time in the past. How can we leverage the tools and capabilities we have today, merging them with the ingenuity of the past, to decrease the time it takes to get to a prototype vehicle, and at the same time increase the quality of our designs to ensure the final product meets expectations?

Despite their origins to create engaging and immersive experiences, commercial off-the-shelf (COTS) game engines are also a powerful tool for rapid development and experimentation. Games and other real-time simulation applications share a common goal: they seek to create approximations of not only human-created designs, but also our natural world, with its rules and constraints. The main distinction is the purpose. For games: to entertain and delight; for real-time simulation applications: to analyze, educate, simulate, and model.

Game engine-based simulation is one of the ways to leverage the spirit of the past: quickly creating an experience in the design before all the decisions are made. Throughout this paper, we will discuss the integration of simulation techniques at every stage of design, from the initial proposals through the final design and delivery; faster and with a better product.

While the development platform which will be the basis of our discussion is Unreal Engine (UE, Unreal), many of the topics and approaches described may (with some effort) be equally applicable to other game engines, such as Unity, Godot, or CryEngine.

We will discuss the experiences, approaches, and abstractions engineered by Offworld Defence Simulations (ODS) within the context of simulating arbitrary vehicles (both crewed and unmanned) in a networked, multiplayer environment. Paired with GS Engineering's experience in vehicle design and systems integration, we present a holistic approach to the merger of simulation and design activities.

ODS specializes in not only creating defense-oriented research, training, and modeling software in Unreal but also in supporting military and industrial partners attempting to do the same. Many of its key personnel came from a game development background and collectively they now focus on satisfying an under-fulfilled need for a readily consumable, modular, and source-available combined-arms military simulation middle-ware framework.

GS Engineering specializes in full vehicle design, subsystem development, and systems integration for both the military and commercial industries. The turn-key engineering services include program management, mechanical and electrical design, software development, intelligent systems, technology integration, weight and cost reduction, survivability, obsolescence engineering, advanced materials, dynamic modeling, analysis, simulation, and laboratory and field testing, through the full product lifecycle to bring innovation to life.

2. GAME ENGINES

Much of the utility of game engine frameworks arise not just from the obvious focus on high-fidelity simulation of real-life phenomena, but also on their ability to coalesce into a vertically integrated development environment which would have, not long ago, required the often-onerous assembly of several disparate systems. Therefore, one of the main value

propositions of using such software packages is economy of effort. Game engines minimize the investment of valuable time and budget into mechanisms that support but do not directly deal with the specific problem-space of interest. Application developers can leverage built-in facilities, which are usually not only more well-understood and documented but also robustly tested through an extensive user community.

As a result of these game engines' "batteries included" philosophy, it is not uncommon to have access to some or all the following sub-systems:

- Audio
- Networking
- Input handling
- Entity management
- Animation
- Rendering
- Artificial intelligence
- Physics

Figure 1 Unreal Engine Interface

Our experience has shown that applications developed with engines such as Unreal (as opposed to in-house implementations) require less time to debug, as common problems are usually able to be resolved by taking advantage of a wealth of community knowledge. Likewise, due to the benefit of years of development and hundreds of developers' attention, these engines support a wide variety of use-cases and address more esoteric edge-cases. By exposing building blocks appropriate for different audiences, development can be made accessible to a wider spectrum of roles, from non-technical product managers and content creators to programmers and engineers. Low-level logic can be exposed for up-stack consumption via data-driven configuration or higher-level scripting.

COTS game engines, particularly those for which source code is available (such as Unreal), benefit from a very broad community of developers and customers. This accelerates feature development, quality assurance, and performance engineering. This serves to maintain a high bar with respect to performance; games are sensitive to performance considerations. A focus on performance can yield the ability to render more frames per unit time or do more things within one frame while maintaining a satisfactory and convincing frame rate.

Developers can request new features and can even "scratch their own itch" and then provide the changes back for inclusion in Unreal itself.

2.1. Bravo Framework for Unreal

ODS created Bravo as a networked, multiplayer environment to which users can connect to participate in military operations. It is also the collective set of constituent modules which make that possible. Major elements of Bravo include everything from dismounted characters, vehicles, artillery fire missions, canned scenarios, projectiles, and geo-anchored MGRS topographic maps with blue force tracking. Bravo was created in part with a desire to be a middleware layer with which developers could rapidly spin up to facilitate the high-level creation approximations of industrial and combat systems, often with no new code being written.

Unreal's built-in vehicles support a modest set of wheel, engine, and transmission parameters. However, out of the box, its control mechanisms are limited to receiving

steering and throttle inputs from a single possessing player. ODS Bravo addresses the limitations by building more capability on top of Unreal's base functionality.

Some basic high-level requirements informed the development of the Bravo framework:

- Multi-passenger
- Multiple usable points of view per role (as a vehicle commander: turned-out, CITV sensors, and third-person cameras).
- Allows shared use of vehicle systems.
- Allows remote control vehicles (UCAV, UGV, Drones).
- Data-driven configuration and composition of vehicles.
- Easy on local and remote network resources.
- The granularity of abstraction feels comfortable to use and must not require undue amounts of boilerplate code.
- Allows for un-manned remote control.
- Allows for the composition of very different types of vehicles:
- Tanks, aircraft, drones.
- Vehicle subsystems can be created once and reused multiple times (i.e., hatches, weapons, lighting, turrets, power train).

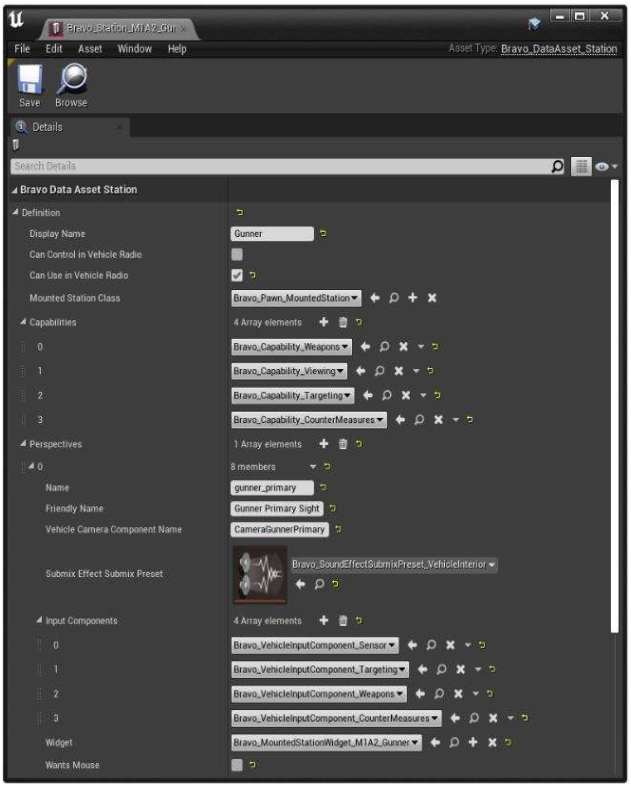

Figure 2 Bravo Settings Panel

One foundational pillar around which the vehicle architecture system was designed is the ability for designers to express their ideas via configuration, in many cases obviating the need for programmer involvement. This allows for valuable programmer time to be spent creating new modules, rather than proliferating duplicated boilerplate code.

Once a piece of the core functionality has been developed, it can be used multiple times via configuration-based composition. At a high level, this is primarily achieved through Unreal's ActorComponents, and hierarchical configuration and data assets referenced within them.

3. SIMULATION ALONGSIDE DESIGN

The capabilities of game engines can be leveraged for many tasks that are done during the development of new vehicles and systems. The possibilities are nearly endless; however, the objective of this paper is not to show how one could create groundbreaking content – that we will leave to the industry. Instead, we delve into unique capabilities that

can be leveraged to visualize designs, increase confidence early in the development effort, increase the value of reviews, and provide the framework for systems integration labs (SILs) for hardware and software development.

3.1. Early Concept Visualization

Early concept development can be difficult. There are often multiple parties involved trying to determine the best way to solve a particular problem set. Once a concept is agreed upon, it can be difficult to communicate the solution to vested parties. This is exacerbated when developing proposal content for open solicitations or competitive bids where proposal writers are firewalled from proposal reviewers. There are a few key problems here that can be addressed using game engines.

Figure 3 shows a vehicle concept generated for a proposal effort. The concept was generated for a proposal to show how GS Engineering would address the specific problem set that was being targeted. This CAD model was developed quickly to support the proposal development, which means components are not real, and this is not a producible design but created for visualization only. Generating this concept with CAD tools enables the proposal effort to be carried forward as a starting point for detailed design.

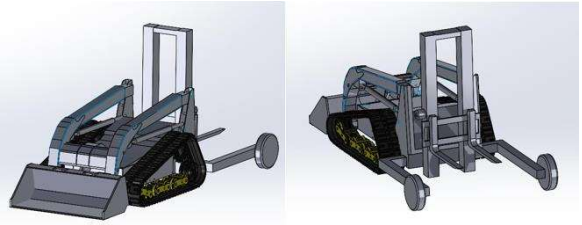

Figure 3: Cad Concept

Figure 4 shows the same concept now rendered in Unreal Engine. To create this representation the model was exported from SolidWorks and imported into Unreal Engine. The additional effort required in Unreal to convert the design into an actor that could be placed in various scenes in unreal was minimal. The time spent in unreal was largely applying realistic textures and finishes to make the vehicle appear to be real. The scenes and vehicle templates that can be used to show the vehicle in operation are nearly endless and available from the Unreal Community.

Figure 4: Proposal Concept

Regardless of how well organized a proposal team is, proposals take time, and a successful proposal needs to stand out above the competition to win an award. Through the development of this proposal, GS Engineering was able to create CAD that would carry forward to detailed design, and with little effort were able to create some stunning visualizations that formed the basis of the proposal which was ultimately selected for award.

3.2. Design Review

Detailed design is when the engineers are doing great work and solving difficult problems. This is also the time that complex designs can change drastically from original concepts. It is important to keep leaders, customers, and stakeholders up to speed throughout the design process. This is another area where game engines provide an edge over classical CAD tools, allowing the concept to be seen in the environment and experienced more fully than traditional CAD tools allow.

3D renderings provide an excellent way to show a preview of what the final design will look like to the non-technical audience, but also for engineers to see how design

decisions impact the final product. There are many options to use for 3D renderings, including many advanced capabilities supported by CAD tools. Game engines provide much of the same capabilities as CAD tools to provide rendering capabilities, however, the capabilities embedded in the game engines open many advantages to taking this extra step.

For example, physics can be applied to dynamic models to show motion, models can be placed in the world to add realism to the models that are being reviewed, vehicle templates can be leveraged to drive vehicles in a simulated environment, and virtual reality/augmented reality (VR/AR) models can be used to show full-scale vehicles.

VR capabilities that exist today provide a stunningly realistic experience. When creating a new vehicle concept, all the modeling in the world with mannequins does not replace the feel of stepping into a vehicle, reaching for controls, and looking at sightlines and obstructions. By leveraging Unreal Engine, the 3D CAD can be imported to a VR template to allow designers and reviewers alike to experience the vehicle. Users can experience what is like to sit in the seat, stick your head out a window, walk around the outside, or even climb on top of a vehicle. This is a surreal experience and allows for virtual hands-on testing of the vehicle concept from the time a basic model is developed. Additionally, design review attendees can be located in geographically disparate locations and meet in an online virtual studio.

Figure 5: VR Review

3.3. Hardware Integration

"By-wire" is a term used in industry to identify an electromechanical subsystem that uses electronic controls to affect a physical system; such as steer-by-wire, brake-by-wire, or shift-by-wire. More and more it is assumed that systems are connected "by wire", and this largely goes unnoticed by users. The advantages are clear when it comes to systems integration, autonomy, and shared vehicle controls. This, however, both increases the complexity of systems integration and opens a host of additional control options for the platforms.

Many development efforts already include the creation of purpose-built test equipment along with the design for hardware and software validation testing. Developing the simulation platform in concert with the detailed design opens a clear opportunity to leverage simulation for the integration and development of control systems. Game engines are designed to work with several different control systems to remain agnostic to the platform. For vehicles in simulation, this control can happen with COTS game controllers, keyboard and mouse, or prototype hardware controllers.

Figure 6: Vehicle Controller

By integrating the hardware with existing simulation assets, a new piece of hardware can be integrated with the simulation before the vehicle design team touches their mouse or keyboard. GS Engineering, as part of a recent development effort, integrated a new control system for vehicle control using an existing test station with surrogate simulation assets. Controls were mapped in software to the functions of a simulated vehicle to allow controlling the simulated vehicle with the proposed vehicle controller. Designers were able to control simulated vehicles with the new controller. They drove simulated vehicles with the controls to get a feel for how they would be used in a vehicle. Designers were able to identify minor issues with the wheel center dead band and adjusted the pedal pressure. This could be done because of the additional seat time using the controls for their intended functions instead of testing the pure electromechanical nature of the subsystem. In a traditional vehicle design effort, these issues would not have been identified until prototype vehicle testing which would have created costly last-minute changes to the design.

Whether the design task is to develop controls or development of a fully integrated control system, the simulation allows this work to be done in parallel with vehicle and test system development using game engines for simulation. This effort proved to be a valuable addition and averted costly change orders during prototype testing.

3.4. Autonomy Development

Fully autonomous vehicles will require hundreds of millions, if not billions of miles to demonstrate their reliability. This would potentially take hundreds of years based on the existing fleet of self-driving cars. Many companies developing self-driving car technology have therefore resorted to simulation to achieve the required levels of reliability. Waymo has been using simulated environments to train, test, and validate its "Waymo driver" software. It has acquired 5 billion simulated miles. Challenging situations encountered in the real world can be targeted in simulation to master edge cases that are less likely in real-world situations [3].

Unreal has also been used to create simulators that allow passengers to provide feedback on their interactions with a self-driving vehicle. Passengers' comfort levels can be quickly tested in multiple configurations [4]. Determining user interaction with autonomous systems early in the development process will increase the speed of development.

US Army developers recently used Unreal to develop and validate autonomy capability, through a framework consisting of sensor and physics simulation, enabling autonomy-in-the-loop. In this case, the software under test was harnessed to a world simulation and interacted with the world by receiving input and providing outputs to the vehicle platform that existed only in simulation [5].

3.5. Digital Prototype

The use of Unreal Engine throughout the lifecycle development effectively supports the tenants of digital engineering. Specifically: producing an authoritative digital model that accurately represents design concepts without the time and expense of building a physical prototype. Rapid iteration of multiple concepts can be evaluated in a fraction of the time and cost required when physical builds are required. This also supports Agile methodology;

quickly implementing additional features and obtaining user feedback for various use cases.

In addition to reducing the number of physical prototypes required to accurately quantify performance, the effective use of digital engineering allows manufacturing and assembly to be effectively simulated: AR/VR can assist in the physical layout of production facilities. AR, in particular, can assist with assembly instructions by overlaying virtual representations of parts onto the systems they are being assembled to. This results in lower task times and higher quality.

3.6. Simulation

Thus far in this paper, we have been discussing the advantages of using game engines for design efforts without yet talking about simulation; the advantages of game engines for simulation almost speak for themselves and open up capabilities that were previously only available for large primes with large budgets.

Unreal with Bravo allows a networked multiplayer environment to which users can connect from around the world to participate in the simulation. Assets can be brought into the Unreal Engine and connected, weather and terrain can be modified to present problem sets, and scenarios can be constructed to test the functions of vehicles. Reviewers from around the world can join the simulation to discuss the vehicle, walk around in a virtual space and experience the systems and subsystems before the design is complete and decisions are set in stone.

Figure 7: Simulation

Simulation facilitates hardware-in-loop or software-in-loop (HIL/SIL) development and validation of hardware and software components alike, whether in a large integrated system or a small one-component system. Whole systems can be simulated within the game engine, down to the component level, for interface with a component under development. Conversely, a single system component can be modeled within the game engine and provided an interface to a whole working system. A particular instance of the latter can be a vehicle platform, where the game engine handles localizing the vehicle in the world, physics interaction with terrain, vehicle-to-vehicle interface (V2V), etc., while the vehicle platform is interfaced and takes its cues from those world parameters from the engine itself. In short, the game engine provides the world, and the system operates in it through a transparent interface.

4. CONCLUSIONS

Game engines are a very attractive platform for rapid application development. There are many practical applications for which an off-the-shelf engine is particularly well-suited. Before any metal is bent, we have the ability to quickly and experimentally develop a solid foundational understanding of the effectiveness of the system we eventually produce.

Data-gathering from simulation sessions is valuable both in automated cases, and ones

where a human is in the loop. Through data-gathering and analysis (via, for instance, analytics events emitted during an experimental simulation session), we have the ability to pose thoughtful questions, and then seek to formulate experiments to answer those questions. This can be a mechanism as simple as new-line delimited JSON, or a complete analytics package, such as ElasticSearch.

In situations with human operator involvement, (for instance, for the purposes of rehearsal), data-gathering and iteration go hand in hand in shaping the design of systems in development. Use-cases can range from testing task saturation of operators using proposed systems, to experimentally weighing trade-offs in terms of costs and feature-set.

4.1. Relationship with other functions

This paper was focused on an engineering perspective of how the utilization of gaming engines can accelerate design activities. This effort requires skillsets that are parallel to traditional design activities to create models that are suitable for simulation. Cad models will be converted into 3d-formats like OBJ and FBX not traditionally used by engineers. There may be additional tools and training necessary to facilitate this capability. However, engineering is not the only function that is starting to see the use of these models. Training platforms are transitioning to VR/AR models to produce an immersive training environment to show instead of telling users how the system functions. VR/AR work instructions are leveraging these models for showing how to do things ranging from an oil change to an engine overhaul. Marketing departments are asking for these models as well to create promotional videos, scale models, and user demonstrations. By bringing this effort into the engineering functions we can both maintain an accurate single source of truth, but also ensure that models shared across functions are consistent and maintained reducing the workload of disparate functions.

4.2. What's next?

Game engines like Unreal provide capabilities for designers that were cost-prohibitive in the past. We are seeing AR and VR enter the engineering design cycle and seeing more fluid connections between design and visualization tools. So, what is next? The gaming industry is going to continue to advance, with more realistic physics engines. What do we have to look forward to in the future for digital design?

Game engines are evolving beyond the "simple" game functionality into tools offering more advanced design and engineering capabilities:

- Replacement/augmentation of physics simulations
- Load Case Development to inform analysis
- Digital Twins
- Integrating Unreal for in-vehicle displays
- Simulation of systems/subsystems in HIL/SIL environments for autonomy development
- AR maintenance and training job aids

It's clear that game engines like Unreal, are positioned for growth and adoption as serious engineering tools going forward.

5. REFERENCES

[1] "DEVELOPMENT OF THE WORLD WAR II JEEP" Accessed on: March 19,

2022. [online] Available: http://nusafm.org/Willys-Jeep.aspx

[2] A. Carter, *The story of the Bantam Jeep*, Accessed on: March 19, 2022. [online] Available: https://www.m201.com/bantam.htm

[3] Andrew Hawkins, "*WELCOME TO SIMULATION CITY, THE VIRTUAL WORLD WHERE WAYMO TESTS ITS AUTONOMOUS VEHICLES"* Accessed on April 4, 2022 [online] Available: https://www.theverge.com/2021/7/6/22565448/waymo-simulation-city-autonomous-vehicle-testing-virtual

[4] Sebastien Loze, *"Autonomous vehicle development and training meets user experience a VI-grade",* Accessed on April 4, 2022 [online] Available: Autonomous vehicle development and training meets user experience at VI-grade - Unreal Engine

[5] John Brabbs, Benjamin Haynes, Thomas Stanko, "Using A Gaming Engine for Autonomous Vehicle Modeling and Simulation", In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 11-13, 2020 [online] Available: http://gvsets.ndia-mich.org/documents/MS2/2020/MS2_1110_Using%20A%20Gaming%20Engine%20for%20Autonomous%20Vehicle%20Modeling%20and%20Simulation_Paper.pdf